trackeR

Infrastructure for Running and Cycling Data from GPS-Enabled Tracking Devices in R

Monika Chudek
Anna Gierlak

Czym jest trackeR?

Pakiet przeznaczony dla osób uprawiających sport (głównie biegaczy i kolarzy). Pomaga analizować dane czasoprzestrzenne zebrane podczas treningów.

#sports #tracking #workCapacity #running #cycling #distributionProfiles

Jakie problemy rozwiązuje pakiet trackeR ?

Wczytywanie danych

- Importuje *sportowe dane* pobrane z urządzeń GPS
- Porządkuje je względem poszczególnych sesji treningowych

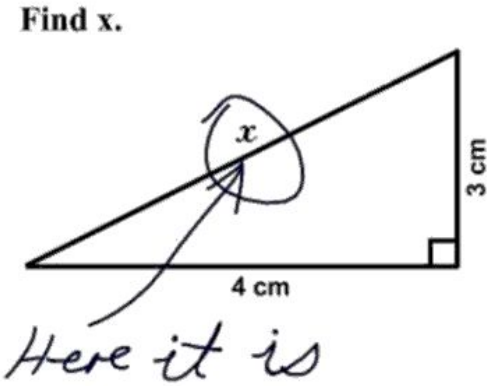
Function	Class	Description
`readTCX()`	TCX file	Read TCX file
`readDB3()`	DB3 file (SQLite)	Read DB3 file
`readJSON()`	Golden Cheetah's JSON file	Read JSON file
`readContainer()`	TCX/DB3/JSON file	Read a TCX/DB3/JSON file
`readDirectory()`	TCX/DB3/JSON files	Read all TCX/DB3/JSON files in a directory

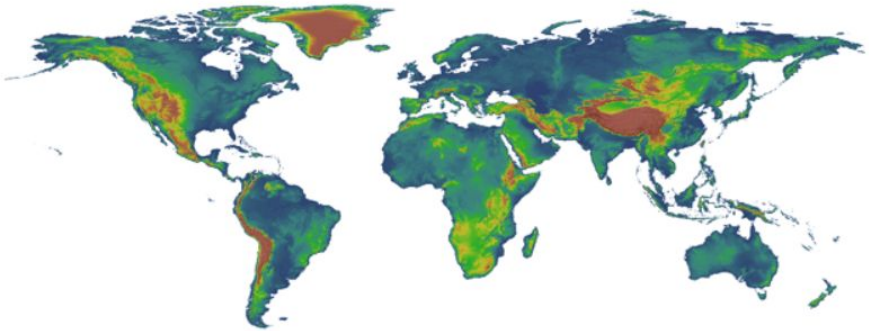
Strukturyzacja danych - klasa *trackeRdata*

```
trackeRdata(dat, units = NULL, cycling = FALSE, sessionThreshold = 2,
correctDistances = FALSE, country = NULL, mask = TRUE, fromDistances = TRUE,
lgap = 30, lskip = 5, m = 11)
```

Poprawka na zmiany wysokości

`correctDistances = FALSE, country = NULL, mask = TRUE`





Imputation process

`lgap = 30, lskip = 5, m = 11`

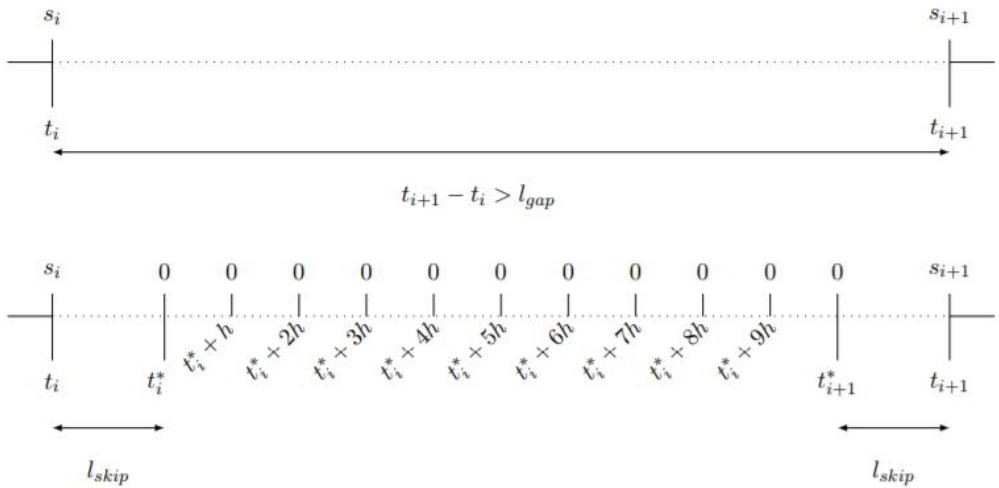


Illustration of the imputation process for speed with $m = 11$.

Demo Code

```
install.packages("trackeR")

library("trackeR")

data("runs", package = "trackeR")

class(runs)

runs[1]
```

Podsumowanie danych

```
summary(runs, session = 1:2,  movingThreshold = 1)
```

Średnia wartość V dla całego treningu (sesji)
Average V

$$\sum_i v_i \frac{\Delta_i K_i}{\sum_i \Delta_i K_i}$$

Średnia wartość V w ruchu
Average V moving

$$\sum_i v_i \frac{\Delta_i K_i I(s_i > s^*)}{\sum_i \Delta_i K_i I(s_i > s^*)},$$

K_i = 1 gdy wartość v_i jest znana, 0 wpp, Δ_i = t_i - $t_{i-1,}$ s^* - movingThreshold.

Wizualizacja danych DEMO

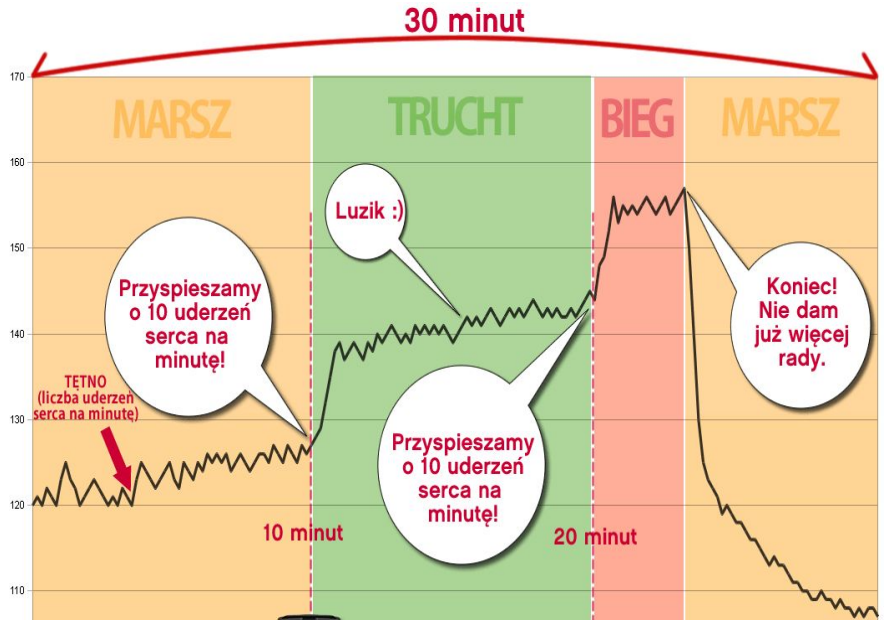
```
plot(runs, session = 27, what = c("altitude", "pace"))
```

```
plotRoute(runs, session = 4, zoom = 13)
leafletRoute(runs, session = 8:13)  # mapa interaktywna
```

```
runSummaryFull <- summary(runs)
plot(runSummaryFull, group = c("total", "moving"),
     what = c("avgSpeed", "distance", "duration", "avgHeartRate"))

timeline(runSummaryFull)
```

Czas spędzony w określonych strefach




```
runZones <- zones(runs[1:4], what = "speed", breaks = list(speed =
                  c(0, 2:6, 12.5)))
plot(runZones)
```

Energia sportowca

P < CP	CP	P > CP
W' ↗ (*replenishing of work capacity*)	moc krytyczna (*critical power*) - maksymalne tempo, które może być utrzymywane bez wyraźnego zmęczenia	W' ↘ (*depletion of work capacity*)

W' - ilość energii, jaka może być wykorzystana do poruszania się w tempie większym od CP (*W' balance*)
W'_0 - W' na początku treningu
W'_0-W' - zużyta energia (*W' expended*)

Energia sportowca

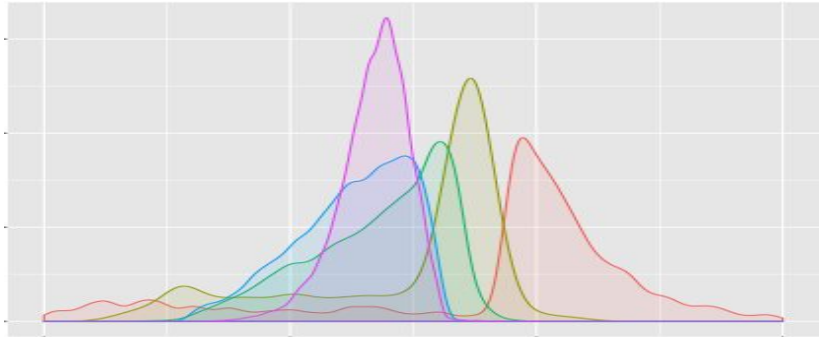
```
wexp <- Wprime(runs, session = 11, quantity = "expended",
               cp = 4, version = "2012")
plot(wexp, scaled = TRUE)
```

Charakterystyka treningu - rozkład p-stwa - *distribution profile*

$$\Pi(v) = \int_0^{t_n} I(v(t) > v)dt$$

```
dProfile <- distributionProfile(runs, session = 1:4,
  what = c("speed", "heart.rate"),
  grid = list(speed = seq(0, 12.5, by = 0.05), heart.rate = seq(0, 250)))
plot(dProfile, multiple = TRUE)
```

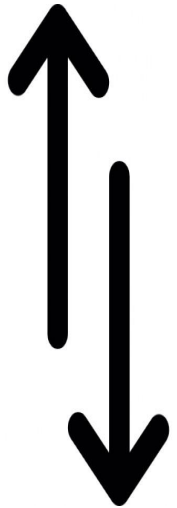
Charakterystyka treningu - rozkład p-stwa - *concentration profile*



```
cProfile <- concentrationProfile(dProfile, what = "speed")
plot(cProfile, multiple = TRUE)
```

Zamiana jednostek

Measurement	Unit(s)
Latitude	Degrees (`degree`, default)
Longitude	Degrees (`degree`, default)
Altitude	Meters (`m`, default), kilometers (`km`), miles (`mi`), feet (`ft`)
Distance	Meters (`m`, default), kilometers (`km`), miles (`mi`), feet (`ft`)
Speed	Meters per second (`m_per_s`, default), kilometers per hour (`km_per_h`), feet per minute (`ft_per_min`), feet per second (`ft_per_s`), miles per hour (`mi_per_h`)
Cadence	Steps per minute (`steps_per_min`, default for running), revolutions per minute (`rev_per_min`, default for cycling)
Power	Watts (`W`, default), kilowatts (`kW`)
Heart rate	Beats per minute (`bpm`, default)
Pace	Minutes per kilometer (`min_per_km`, default), minutes per mile (`min_per_mi`), seconds per meter (`s_per_m`)
Duration	Seconds (`s`), minutes (`min`), hours (`h`) – default is the largest possible unit for which the duration is larger than 1



Zamiana jednostek

```
getUnits(run)
```

```
runTr2 <- changeUnits(runs, variable = "speed", unit = "mi_per_h")
getUnits(runTr2)
```

```
m_per_s2ft_per_h <- function(x) x * 3937/1200 * 3600
changeUnits(summary(runs, session = 1), variable = "speed", unit =
"ft_per_h")
```

Sprawdzanie poprawności danych

```
install.packages(“gridExtra”)
library(gridExtra)

plot1 <- plot(runs, session = 4, what = "speed", threshold = FALSE)
run4 <- threshold(runs[4], variable = "speed", lower = 0, upper = 12.5)
plot2 <- plot(run4, what = "speed", threshold = FALSE) +
                ggplot2::expand_limits(y = c(0, 21))

grid.arrange(plot1, plot2, ncol=2)
```

Wygładzanie - metoda okna ruchomego

```
run4S_20 <- smoother(run4, what = "speed", fun = "median", width = 20)
plot3 <- plot(run4S_20, what = "speed", smooth = FALSE) +
                ggplot2::expand_limits(y = c(0, 12.5))

run4S_5 <- smoother(run4, what = "speed", fun = "median", width = 5)
plot4 <- plot(run4S_5, what = "speed", smooth = FALSE) +
                ggplot2::expand_limits(y = c(0, 12.5))

grid.arrange(plot3, plot4, ncol=2)
```

Dziękujemy za uwagę!

Journal of Statistics:
trackeR: Infrastructure for Running and Cycling Data from GPS-Enabled Tracking Devices in R